Decision Trees & Random Forests.

# Introduction

* Decision trees are suitable for instances where a number of observed factors can be used to predict an outcome.
* Decision trees have nodes (split for the value of certain attributes) and edges (outcome of splitting a node).
* The node that performs the first split is known as a root node.
* Terminal nodes are the final nodes that predict the outcome.
* Entropy and information gain are the mathematical methods of choosing the best split. It is necessary because not all observed parameters will predict the outcome accurately
* The biggest con of decision trees is that they don’t tend to have a lot of accuracy due to high variance. This can be improved with random forests which is an improved bagging method.
* Bootstrapped sample is sampling from a training set with replacement.
* For classification, every time a split is considered, a random sample (m) is chosen from the full set of features (p). M is usually chosen as the squared root of p. a new random p is chosen for every split.
* Random forests help us avoid averaging highly correlated quantities which does not significantly reduce variance and can arise if bagging was used.

# Decision trees and Random Forests in python

## Decision trees

* Data cleaning
* EDA
* Set X to features and y to target
* Train test split with train\_test\_split from sklearn.model\_selection
* From sklearn.tree import DecisionTreeClassifier
* Instantiate the decision tree classifier
* Train (fit) the model with train data set
* Predict with the model using the testing data set
* Import classification report and confusion matric from sklearn.metrics
* Print the classification report and confusion matrix respectively

## Random forests

* From sklearn.ensemble import RandomForestClassifier
* Instantiate the random forest classifier
* Fit (train) the model
* Run the predictions
* Print the confusion matrix and classification report.